Demo Abstract: RPiaaS: A Raspberry Pi Testbed for Validation of Cloud Resource Management Strategies

Pieter-Jan Maenhaut*†, Bruno Volckaert†, Veerle Ongenae* and Filip De Turck†
*Ghent University, Faculty of Engineering and Architecture, Dept. of Information Technology
Valentin Vaerwyckweg 1, 9000 Ghent, Belgium
†Ghent University – imec, IDLab, Dept. of Information Technology, iGent
Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium
Email: pieterjan.maenhaut@ugent.be

***Abstract*—With cloud computing, efficient resource management is of great importance, as it has a direct impact on the scalability of the cloud application, and can result in significant energy and cost reductions. In recent years, a lot of research has been done regarding the management of cloud resources, resulting in multiple novel resource allocation strategies. Validation of these strategies however is often only based on simulations, as large experiments using real cloud infrastructure are both expensive and time-consuming.**

In this demo we present RPiaaS, a low-cost and energy-efficient cloud testbed built using Raspberry Pi's. The testbed provides an easy-to-use environment for the initial evaluation of novel cloud resource management strategies, and is designed to facilitate the step from simulations towards experimental evaluations on larger cloud testbeds.

I. Introduction

Cloud applications can greatly benefit from efficient resource management. Efficient resource usage will result in higher scalability, as there is a more optimal usage of the available resources. At the same time, the energy footprint can be reduced as fewer instances need to be provisioned, resulting in lower operating costs. Over the recent years, a lot of research has been done regarding the efficient allocation of cloud resources [1], [2]. Evaluation is however often only done using simulations, for example by using CloudSim [3]. Although simulation is an important tool for the development of new protocols and algorithms for cloud resource management, experimental evaluations using real hardware are essential for the validation of new research ideas as these often result in new insights. The evaluated experimental setup could for example introduce additional hardware constraints which were not taken into account in the design of the new algorithm or the simulations or it could be used to more accurately tune the simulation parameters.

Running experiments on the public cloud however is both costly and time-consuming. This is especially true for the design and fine-tuning of new resource allocation strategies, as these often require multiple incremental iterations of experiments using multiple cloud instances. When the executed experiments fail during the execution, for example due to hardware constraints or a faulty algorithm, these experiments can become very costly. In this demo we present Raspberry Pi as a Service (RPiaaS), a low-cost cloud testbed built on top of Raspberry Pi 3 boards, which was designed to facilitate the step towards experimental evaluations. RPiaaS provides an easy-to-use interface for both the provisioning and monitoring of the different types of cloud resources.

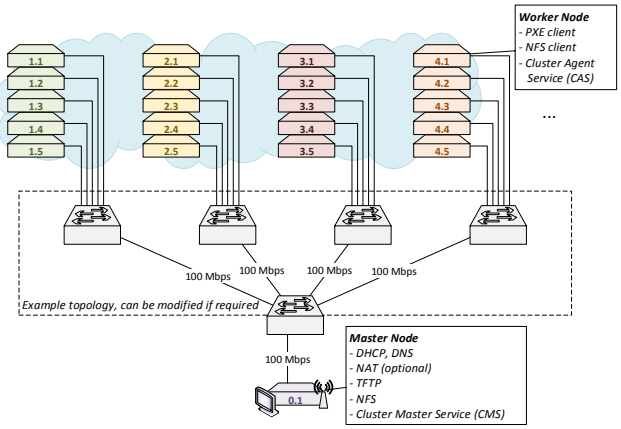

Fig. 1. Overview of the RPiaaS testbed for validation of cloud resource management strategies.

II. Demo Setup

Figure 1 provides an overview of the different components of the RPiaaS testbed. The testbed consists of multiple worker nodes, aggregated in small clusters. Every node in the cluster is interconnected, and there is a master node managing the whole testbed. The master node can be a Raspberry Pi or any device running a Linux distribution. For this demo, we configured a small cluster consisting of 15 worker nodes and 1 master node.

The worker nodes mount the root filesystem through NFS by either doing a network boot (TFTP boot similar to a PXE boot) or by using a minimal image on the memory card. This

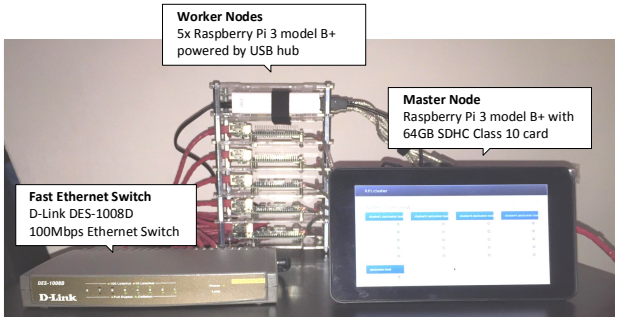

Fig. 2. A fraction of the presented demo setup, consisting of 5 worker nodes.

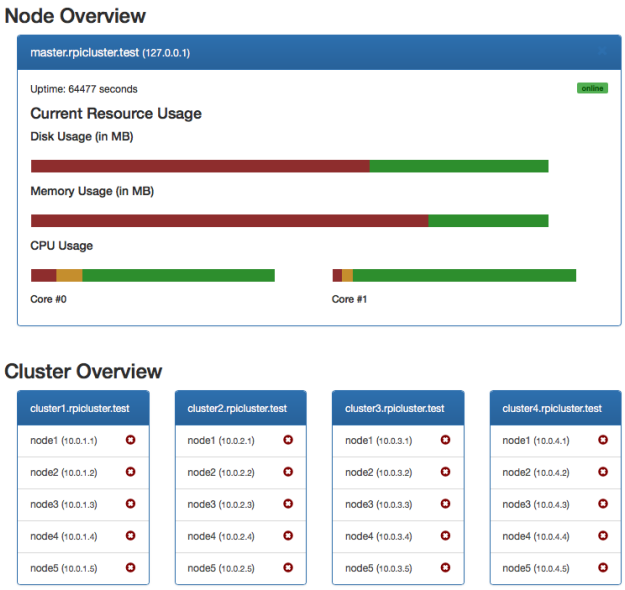

Fig. 3. Partial screen capture of the dashboard provided by the CMS.

allows for easy customization of the operating system and software for the worker nodes, as there is no need to update the individual nodes, for example by copying a new operating system to the memory card. Having to update the memory cards manually can be a cumbersome process, especially for larger setups consisting of a high number of worker nodes.

Every worker node is running the lightweight Cluster Agent Service (CAS) for communication with the master node. The CAS is developed using Node.js and provides a RESTful API towards the different resources for the selected node. The master node is running the Cluster Management Service (CMS), which is developed using a combination of Node.js, Python, HTML, JavaScript and Bash scripting. The CMS provides a RESTful API towards the whole testbed, together with a web-based dashboard. Figure 2 shows a small part of the presented demo setup, whereas Figure 3 presents a partial screen capture of the dashboard provided by the CMS.

The CMS polls all worker nodes using a configured time interval. During this update, the current resource utilization (such as the CPU load, memory and local disk usage) is retrieved from the worker nodes and stored in a central NoSQL database on the master node. This way not only the actual resource usage can be displayed on the management interface, but also the historical usage over time for each worker node individually. Both the CMS and CAS can be easily extended or customized in order to support a wide range of experiments.

III. Main Operations

The following features are currently implemented within RPiaaS, and will be illustrated during the demo.

A. Cluster Administration

New nodes can be easily added to the cluster by modifying a single configuration file on the master node. The configuration for the different services can be automatically generated by the CMS. For the creation of the root file system, a preferred operating system can be installed and configured onto the memory card of a single worker node, together with the CAS service and other software packages required for the experiment. The whole filesystem can then be copied to the master node using *rsync*, a remote and local file synchronization tool for unix systems.

B. Control Experiments

As the public key from the master node is copied to the operating system of the worker nodes, commands can be easily executed on all worker nodes through SSH. Furthermore, the REST API of the CAS includes several POST methods to control jobs on the worker node, but this requires a simple control script (e.g. a bash or python script) implementing the different operations.

C. Resource Usage Monitoring

Both the current and historical resource usage can be viewed through the web-based dashboard of the CMS. This data is also available through the API, or can be retrieved directly from the central database. The CMS and CAS can be easily extended, for example to monitor the current GPU utilization of the nodes, or to include other types of hardware resources.

Acknowledgment

All developed source code will be made available to the general public through GitHub[1], and we encourage fellow researchers within the field to try out and customize the code for their own research projects.

References

[1] Z. Lu, S. Takashige, Y. Sugita, T. Morimura, and Y. Kudo, "An analysis and comparison of cloud data center energy-efficient resource management technology," *International Journal of Services Computing (IJSC)*, vol. 2, no. 4, pp. 32 – 51, 2014.

[2] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567 – 619, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10922-014-9307-7

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011. [Online]. Available: http://dx.doi.org/10.1002/spe.995

[1] https://github.com/IBCNServices/RPiaaS